medial law and functors

Johan G. F. Belinfante
2009 February 12

```
In[1]:= SetDirectory["l:"]; << goedel.09feb11b;<< tools.m

        :Package Title: goedel.09feb11b            2009 February 11 at 2:45 p.m.

        It is now:  2009 Feb 12 at 11:8

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 February 9

        weightlimit = 40
```

summary

A commutative monoid **x** is a functor from **direct[x, x]** to **x**. The key to this is the medial law. A class **x** is **medial** if the following holds: (another name for this concept is **entropic**)

```
In[2]:= medial[x_] := equal[composite[x, cross[x, x]], composite[x, cross[x, x], TWIST]]
```

A binary operation is medial if its elements satisfy the condition **(t · u) · (v · w) = (t · v) · (u · w)**. Note the interchange of the middle pair of elements **u** and **v**. In the **GOEDEL** program, the **TWIST** function accomplishes this interchange. A category is medial if and only if it is commutative.

medial categories

Lemma. Converting **TWIST** to **SWAP**.

```
In[4]:= composite[TWIST, cross[composite[id[cart[u, v]], inverse[SECOND]],
           composite[id[cart[x, y]], inverse[FIRST]]]] // ReifTriNormality

Out[4]= composite[TWIST, cross[composite[id[cart[u, v]], inverse[SECOND]],
           composite[id[cart[x, y]], inverse[FIRST]]]] ==
         composite[SWAP, cross[composite[id[cart[v, y]], inverse[FIRST]],
           composite[id[cart[u, x]], inverse[SECOND]]]]

In[5]:= composite[TWIST, cross[composite[id[cart[u_, v_]], inverse[SECOND]],
           composite[id[cart[x_, y_]], inverse[FIRST]]]] :=
         composite[SWAP, cross[composite[id[cart[v, y]], inverse[FIRST]],
           composite[id[cart[u, x]], inverse[SECOND]]]]
```

Corollary. A special case of the lemma of independent interest.

```
In[7]:= SubstTest[composite, TWIST, cross[composite[id[cart[u, V]], inverse[SECOND]],
          composite[id[cart[V, v]], inverse[FIRST]]], {u → set[x], v → set[y]}] // Reverse

Out[7]= composite[TWIST, cross[LEFT[x], RIGHT[y]]] == composite[SWAP, cross[RIGHT[y], LEFT[x]]]

In[8]:= composite[TWIST, cross[LEFT[x_], RIGHT[y_]]] :=
         composite[SWAP, cross[RIGHT[y], LEFT[x]]]
```

Lemma. A simplification rule.

```
In[10]:= Assoc[cat[x], id[cartsq[range[cat[x]]]], SWAP]

Out[10]= composite[cat[x], SWAP, id[cart[range[cat[x]], range[cat[x]]]]] ==
          composite[cat[x], SWAP]

In[11]:= composite[cat[x_], SWAP, id[cart[range[cat[x_]], range[cat[x_]]]]] :=
          composite[cat[x], SWAP]
```

Theorem. Medial categories are commutative.

```
In[12]:= SubstTest[implies, equal[u, v], equal[composite[u, w], composite[v, w]],
           {u -> composite[cat[x], cross[cat[x], cat[x]]],
            v -> composite[cat[x], cross[cat[x], cat[x]], TWIST],
            w → cross[composite[id[cart[ids[cat[x]], V]], inverse[SECOND]],
              composite[id[cart[V, ids[cat[x]]]], inverse[FIRST]]]}] // Reverse

Out[12]= or[equal[cat[x], composite[cat[x], SWAP]],
           not[equal[composite[cat[x], cross[cat[x], cat[x]]],
             composite[cat[x], cross[cat[x], cat[x]], TWIST]]]] == True

In[13]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma. Commutative categories are medial.

```
In[14]:= SubstTest[implies,
           and[associative[t], equal[t, flip[t]]], medial[t], t → cat[x]] // Reverse

Out[14]= or[equal[composite[cat[x], cross[cat[x], cat[x]]],
            composite[cat[x], cross[cat[x], cat[x]], TWIST]],
           not[equal[cat[x], composite[cat[x], SWAP]]]] == True

In[15]:= (% /. x → x_) /. Equal→ SetDelayed
```

Theorem. A category is medial if and only if it is commutative.

```
In[16]:= equiv[equal[composite[cat[x], cross[cat[x], cat[x]]],
           composite[cat[x], cross[cat[x], cat[x]], TWIST]],
          equal[cat[x], composite[cat[x], SWAP]]]

Out[16]= True

In[17]:= equal[composite[cat[x_], cross[cat[x_], cat[x_]]],
           composite[cat[x_], cross[cat[x_], cat[x_]], TWIST]] :=
          equal[cat[x], composite[cat[x], SWAP]]
```

```
(% /. x → x_) /. Equal→ SetDelayed
```

Corollary. A commutative category is medial. (Restatement without the **cat** wrapper.)

```
In[18]:= SubstTest[implies, and[equal[x, cat[t]], equal[x, flip[x]]], medial[x], t → x] // Reverse

Out[18]= or[equal[composite[x, cross[x, x]], composite[x, cross[x, x], TWIST]],
            not[category[x]], not[equal[x, composite[x, SWAP]]]] == True

In[19]:= or[equal[composite[x_, cross[x_, x_]], composite[x_, cross[x_, x_], TWIST]],
            not[category[x_]], not[equal[x_, composite[x_, SWAP]]]] := True
```

Corollary. (An example: **INTMUL**.)

```
In[20]:= SubstTest[medial, cat[x], x → INTMUL] // Reverse

Out[20]= equal[composite[INTMUL, cross[INTMUL, INTMUL]],
            composite[INTMUL, cross[INTMUL, INTMUL], TWIST]] == True

In[21]:= composite[INTMUL, cross[INTMUL, INTMUL], TWIST] :=
          composite[INTMUL, cross[INTMUL, INTMUL]]
```

Corollary. A group is medial if and only if it is abelian.

```
In[23]:= SubstTest[medial, cat[t], t → gp[x]] // Reverse

Out[23]= equal[composite[gp[x], cross[gp[x], gp[x]]],
            composite[gp[x], cross[gp[x], gp[x]], TWIST]] == equal[composite[gp[x], SWAP], gp[x]]

In[24]:= equal[composite[gp[x_], cross[gp[x_], gp[x_]]],
            composite[gp[x_], cross[gp[x_], gp[x_]], TWIST]] :=
          equal[composite[gp[x], SWAP], gp[x]]
```

---

commutative categories are functors

Lemma.

```
In[25]:= ImageComp[cat[x], id[id[ids[cat[x]]]], V] // Reverse

Out[25]= image[cat[x], id[ids[cat[x]]]] == ids[cat[x]]

In[26]:= image[cat[x_], id[ids[cat[x_]]]] := ids[cat[x]]
```

Theorem.

```
In[27]:= ImageComp[cat[x], id[domain[cat[x]]], cartsq[ids[cat[x]]]]

Out[27]= image[cat[x], cart[ids[cat[x]], ids[cat[x]]]] == ids[cat[x]]

In[28]:= image[cat[x_], cart[ids[cat[x_]], ids[cat[x_]]]] := ids[cat[x]]
```

Lemma. (A direct consequence of the definition of functor.)

```
In[29]:= (SubstTest[implies, and[FUNCTION[t], equal[domain[t], range[u]], equal[composite[t, u],
              composite[v, cross[t, t]]], subclass[image[t, ids[u]], ids[v]]],
            functor[t, u, v], {u → direct[t, t], v → t}] // Reverse) /. t → cat[x]

Out[29]= or[functor[cat[x], composite[cross[cat[x], cat[x]], TWIST], cat[x]],
           not[equal[cart[range[cat[x]], range[cat[x]]], domain[cat[x]]]],
           not[equal[cat[x], composite[cat[x], SWAP]]]] == True

In[30]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary. (An example: **SYMDIF**.)

```
In[31]:= SubstTest[implies, and[equal[cart[range[cat[x]], range[cat[x]]], domain[cat[x]]],
            equal[cat[x], flip[cat[x]]]], functor[cat[x],
            composite[cross[cat[x], cat[x]], TWIST], cat[x]], x → SYMDIF] // Reverse

Out[31]= functor[SYMDIF, composite[cross[SYMDIF, SYMDIF], TWIST], SYMDIF] == True

In[32]:= functor[SYMDIF, composite[cross[SYMDIF, SYMDIF], TWIST], SYMDIF] := True
```

Theorem. (An example: **CUP**.)

```
In[33]:= SubstTest[implies, and[equal[cart[range[cat[x]], range[cat[x]]], domain[cat[x]]],
            equal[cat[x], flip[cat[x]]]],
           functor[cat[x], composite[cross[cat[x], cat[x]], TWIST], cat[x]], x → CUP] // Reverse

Out[33]= functor[CUP, composite[cross[CUP, CUP], TWIST], CUP] == True

In[34]:= functor[CUP, composite[cross[CUP, CUP], TWIST], CUP] := True
```

---

abelian monoids

Every pair of morphisms in a category **x** are composable if its domain is the cartesian square of its range. (This is also equivalent to the statement that the category has a single identity morphism.)

Theorem. If every pair of morphisms in a category **x** are composable, and if x is commutative, then **x** is a functor from **direct[x, x]** to **x**. (This is obtained by eliminating the **cat** wrapper.)

```
In[38]:= SubstTest[implies, and[equal[x, cat[t]], equal[x, flip[x]],
            equal[domain[x], cartsq[range[x]]]], functor[x, direct[x, x], x], t → x] // Reverse

Out[38]= or[functor[x, composite[cross[x, x], TWIST], x],
           not[category[x]], not[equal[x, composite[x, SWAP]]],
           not[equal[cart[range[x], range[x]], domain[x]]]] == True

In[39]:= or[functor[x_, composite[cross[x_, x_], TWIST], x_],
           not[category[x_]], not[equal[x_, composite[x_, SWAP]]],
           not[equal[cart[range[x_], range[x_]], domain[x_]]]] := True
```

Corollary. If **x** is an abelian monoid, then **x** is a functor from **direct[x, x]** to **x**.

```
In[40]:= Map[not, SubstTest[and, implies[and[p2, p3, p4], p5],
           not[implies[and[p1, p2], p5]], {p1 → member[x, MONOIDS], p2 -> equal[x, flip[x]],
            p3 → category[x], p4 -> equal[cart[range[x], range[x]], domain[x]],
            p5 -> functor[x, composite[cross[x, x], TWIST], x]}]] // Reverse

Out[40]= or[functor[x, composite[cross[x, x], TWIST], x],
           not[equal[x, composite[x, SWAP]]], not[member[x, MONOIDS]]] == True

In[41]:= or[functor[x_, composite[cross[x_, x_], TWIST], x_],
           not[equal[x_, composite[x_, SWAP]]], not[member[x_, MONOIDS]]] := True
```

examples

Once the medial law has been derived for a particular abelian monoid, one can simply use **AssertTest** to derive the functor property. Some examples are provided here.

```
In[42]:= functor[NATADD, composite[cross[NATADD, NATADD], TWIST], NATADD] // AssertTest

Out[42]= functor[NATADD, composite[cross[NATADD, NATADD], TWIST], NATADD] == True

In[43]:= functor[NATADD, composite[cross[NATADD, NATADD], TWIST], NATADD] := True

In[44]:= functor[NATMUL, composite[cross[NATMUL, NATMUL], TWIST], NATMUL] // AssertTest

Out[44]= functor[NATMUL, composite[cross[NATMUL, NATMUL], TWIST], NATMUL] == True

In[45]:= functor[NATMUL, composite[cross[NATMUL, NATMUL], TWIST], NATMUL] := True

In[46]:= functor[INTADD, composite[cross[INTADD, INTADD], TWIST], INTADD] // AssertTest

Out[46]= functor[INTADD, composite[cross[INTADD, INTADD], TWIST], INTADD] == True

In[47]:= functor[INTADD, composite[cross[INTADD, INTADD], TWIST], INTADD] := True

In[48]:= functor[INTMUL, composite[cross[INTMUL, INTMUL], TWIST], INTMUL] // AssertTest

Out[48]= functor[INTMUL, composite[cross[INTMUL, INTMUL], TWIST], INTMUL] == True

In[49]:= functor[INTMUL, composite[cross[INTMUL, INTMUL], TWIST], INTMUL] := True
```

abelian groups as functors

Theorem. The case of an abelian group.

```
In[50]:= SubstTest[implies, and[equal[cart[range[cat[t]], range[cat[t]]], domain[cat[t]]],
            equal[cat[t], flip[cat[t]]]],
           functor[cat[t], composite[cross[cat[t], cat[t]], TWIST], cat[t]], t → gp[x]] // Reverse

Out[50]= or[functor[gp[x], composite[cross[gp[x], gp[x]], TWIST], gp[x]],
           not[equal[composite[gp[x], SWAP], gp[x]]]] == True
```

```
In[51]:= or[functor[gp[x_], composite[cross[gp[x_], gp[x_]], TWIST], gp[x_]],
           not[equal[composite[gp[x_], SWAP], gp[x_]]]] := True
```

Corollary. If **x** is an abelian group, then **x** is a functor from **direct[x, x]** to **x**. (Obtained by removing the **gp** wrapper.)

```
In[52]:= SubstTest[implies, and[equal[x, gp[t]], equal[flip[x], x]],
            functor[x, direct[x, x], x], t → x] // MapNotNot // Reverse
Out[52]= or[functor[x, composite[cross[x, x], TWIST], x],
           not[equal[x, composite[x, SWAP]]], not[member[x, GROUPS]]] == True

In[53]:= or[functor[x_, composite[cross[x_, x_], TWIST], x_],
           not[equal[x_, composite[x_, SWAP]]], not[member[x_, GROUPS]]] := True
```